07

Debug

Transcripción

[00:00] Y bien, lo prometido es deuda. Hemos vuelto aquí al código y ahora les voy a enseñar cómo es que podemos ver la ejecución de nuestro método y ver en vivo la pila de ejecución aquí en el código. Comenzamos primero, haciendo doble clic en la línea donde queremos comenzar a ver el proceso.

[00:22] En este caso, como quiero comenzar desde el inicio, aquí en el método main, aquí en la primera línea, System.out.printIn, inicio main, yo voy a hacer doble clic en la línea 5 y automáticamente vemos que aquí generó un punto. Ese punto es lo que vamos a llamar the breakpoint, punto de quiebre. ¿Por qué? Porque lo que vamos a hacer es lo que llamamos debug.

[00:47] El debug es la ejecución de nuestro sistema, pero donde él encuentra este breakpoint él va a parar y nos va a mostrar todo lo que hay en la memoria, en ese momento. Vamos a verlo aquí en práctica. Para eso, para el debug, nosotros tenemos que ir al botón de acá, que es este bichito, debug flujo. Este bichito está al lado de play, del botón de play.

[01:15] ¿Y él qué nos va a decir? Que si queremos ejecutar esta clase en modo debug, le apretamos ahí, y nos pregunta si queremos cambiar la perspectiva Eclipse. ¿Por qué? Porque cuando estamos en modo debug, cuando estamos debugueando una aplicación, Eclipse tiene otras otras ventanas que nos muestran herramientas un poco más útiles.

[01:36] Ahorita estamos en nuestra nuestra perspectiva, digamos de desarrollo, pero cuando es debug, él tiene otras ventanitas con información útil al

respecto, entonces le decimos que sí. Switch. Y vemos cómo cambió toda nuestra perspectiva Eclipse y vemos que aquí dice Flujo.main(String) line 5, y él se detuvo aquí en Inicio de main, y esto que vemos acá, estas tres rayas representan la pila de ejecución.

[02:07] ¿Cómo es que funciona el debug? Tenemos estos botones acá que este es para resumir el debug, o sea para acabarlo, y decirle "sigue con tu camino". Tenemos el botón de stop, que es F2. O lo podemos clicar también stop directamente para decirle: "Detenme aquí el programa, mata el proceso".

[02:29] Tenemos aquí el step in, que es F5. ¿Step in qué significa? Entra a ese pedazo de código y tenemos el step over, que es F6, que es simplemente "pasa a la siguiente línea". Quizás están un poco mareados con esto, pero aquí en la práctica yo estoy seguro que van a agarrar muy rápido la onda.

[02:52] Entonces le vamos a dar aquí a step over, y él hizo un salto de línea y nos llevó a método 1. En método 1, él recibe los argumentos. En este momento no hay ningún argumento entonces no hay nada, es un string vacío. Y él ya imprimió inicio de Main. Y en método 1. ¿Qué es lo que hay? Mira, si le damos otro step over, él va automáticamente a al siguiente System.out.println y él ya ejecutó todo lo demás.

[03:27] ¿Por qué? Porque yo no le dije que entre a explorar método 1. Yo le di un este over. Entonces esa es la diferencia. Como le di step over, él salió de método 1 y método 1 adentro hizo toda su gestión y ahora él solo va a imprimir fin de main, le doy a step over y acabó el proceso y listo, vemos que nuevamente aquí la pila de ejecución ya no está porque ya no hay nada más en la pila en este momento.

[03:53] Vamos otra vez a darle debug, comenzamos aquí. Y esta vez yo le voy a dar el primer step over solo en la primera línea porque en método 1, yo quiero ver qué hay adentro del método 1, yo quiero saber que está ejecutando adentro de método 1. Entonces le voy a dar aquí. Y vemos que automáticamente él ya

llegó a método 1 a inicio de método 1 y mira qué es lo que está a nuestra izquierda.

[04:22] Está abajo main y método 1 aquí encima. Bien parecido a lo que estábamos haciendo aquí en nuestra presentación. ¿Qué hacemos? Step over aquí y vemos que él entra a método 2. ¿Queremos saber qué hay en método 2? Sí, vamos para método 2, llegamos a método 2, y llegamos a método 2 y mira qué sucede aquí a la izquierda. Se apiló método 2 encima. Estamos en este momento de la ejecución del programa.

[04:54] Ahora, ¿qué es lo que va a hacer él? Le damos step over. Él va a ejecutar ese for y él va a imprimir una vez, dos veces, tres veces, cuatro veces y cinco veces, y vemos que aquí tenemos información del índice. Actualmente l5. Con esta perspectiva del debug, nosotros tenemos mucho poder para ver qué está pasando exactamente en el código en este momento.

[05:20] Es muy usado en el trabajo del día a día. Por lo menos yo con estos pasos que son bien básicos, yo me las arreglo en el trabajo y logro encontrar errores en el código, logro saber digamos qué necesita ser re factorado a veces, pero hay muchas más herramientas de debug, incluso aquí en Alura tenemos un curso solo de debug en Eclipse.

[05:42] Es un curso pequeño pero muy bueno para que puedan sacarle el jugo a esta herramienta. Y bueno, continuando aquí con el ejemplo del debug, vemos que ahora él va a seguir ahí. Y como i ya es mayor que 5, él sale del for ¿y qué va a ser? Va a imprimir Fin método 2. Miren, aquí este lado, método 2 sigue encima de método 1. Vamos a darle step over nuevamente. Y él sale de aquí, cierra el método.

[06:13] Entonces, una vez que él sale, mira qué pasó aquí. Método 2 se fue. Y regresa a Fin de método 1. ¿Qué es lo que va a ser él aquí? Imprimir método 1, Fin de método 1 y ¿qué es lo que va a hacer? Salir y volver aquí. Y mira lo que

está imprimiendo aquí, Fin método 1. Y mira la pila, solamente está main. ¿Por qué?

[06:37] Porque ya estamos acabando con toda la ejecución del programa prácticamente. Vamos a darle el último step over, Fin de main, se acabó el proceso. Disconnected, terminated, está aquí. ¿Qué significa el? El servidor de debug ya no está funcionando, está desconectado y el proceso fue exterminado. ¿Por qué?

[06:57] Porque ya salimos del flujo. El flujo ya se completó automáticamente. Se cerró y vimos cómo paso a paso se va pues imprimiendo una línea, otra línea, por dónde es que está yendo el código? Incluso si es que quieren saber cómo funciona la misma librería de Java, por ejemplo, esta librería System.out.println, si tienen curiosidad, pueden hacer esto de aquí.

[07:23] Le dan un step in aquí dentro y miren, los va a llevar a la clase propia de Java, y ahí pueden también ir explorando un poco. Esta es la clase de PrintStream, por ejemplo. Ahí ustedes ya pueden ir explorando cómo es que funciona, Java por dentro, esa es la ventaja del debug, ustedes pueden saber lo que sucede atrás del telón, tanto a nivel de su código, que han escrito ustedes como a nivel mismo del lenguaje.

[07:52] Entonces, esta es una base para depurar código, como les dije antes, acá depuramos códigos, depuramos errores. En el día a día, la habilidad que tengas de buguear un código puede ser un diferencial a la hora de encontrar un error, muchas veces dependiendo de esta habilidad, puedes demorarte o cinco minutos o una hora para encontrar un error en el código.

[08:14] Y muchas veces esto sucede en producción, donde donde el tiempo literalmente es dinero. Les recomiendo bastante practicar esto del debug, no solamente con el código de ustedes, sino también con el mismo código de Java. Y bueno, hemos visto hasta ahora lo que es el mundo ideal. Hasta ahora

ustedes pueden preguntarse: "Diego, no he visto la palabra excepcion por ninguna parte. ¿Qué pasó?"

[08:37] Bueno. Este es el mundo ideal. Les he presentado un flujo completo en el mundo ideal, sin errores, no pasó nada malo, todo salió bien, pero el mundo real no es así. Vamos a ver en la siguiente clase cómo es más o menos un error en medio del código. Vamos a causar un error y vamos a ver cómo podemos lidiar con él. No se pierdan, nos vemos.